<u>Part 3</u>

Lexing

Lexing in a Nutshell

- Convert input text into a token stream

```
           b = 40 + 20*(2+3)/37.5

NAME = NUM + NUM * ( NUM + NUM ) / FLOAT
```

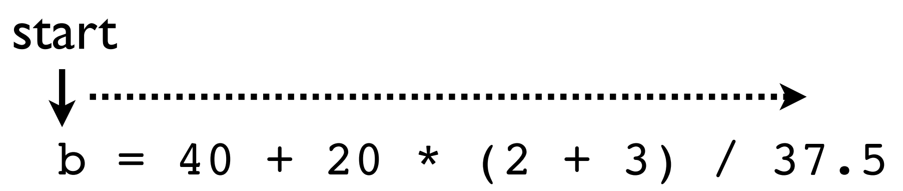
- Token is an object with a type and value

```
b    ⟶   ('NAME','b')

=    ⟶   ('ASSIGN','=')

40   ⟶   ('NUM','40')
```

- Question: How to do it?

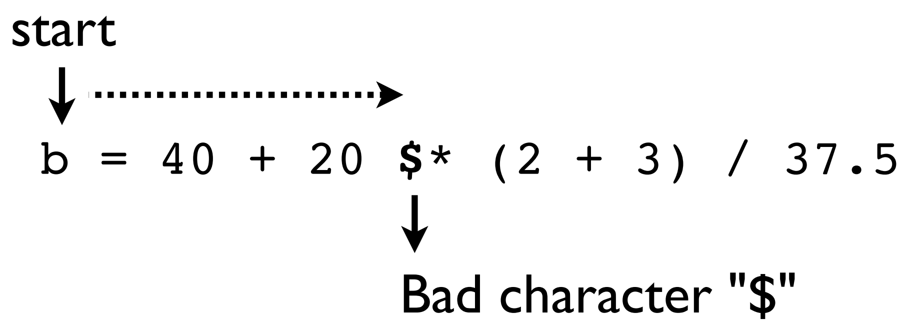
Text Scanning

- Must perform a linear text scan



```
b = 40 + 20 * (2 + 3) / 37.5
```

- ALL characters must be consumed
- Otherwise error:



```
b = 40 + 20 $* (2 + 3) / 37.5
```

Token Matching

- Tokens are formally described by regex

```
NAME   = r'[A-Za-z_][A-Za-z0-9_]*'
NUMBER = r'\d+'
```

- It is not hard to code something by hand
- But few people do this. There are tools that automate it and make it a lot easier (PLY, SLY, PyParsing, ANTLR, etc.)
- Demo:

Commentary

- Tokenizing is NOT an interesting problem in the context of modern compiler writing
- Yes, it is an essential part of parsing.
- But, it's hardly the most important thing.

Project

- Find the file `wabbit/tokenize.py`
- Follow instructions inside